



# SISTEMA BRASILEIRO DE TELEVISÃO DIGITAL

## Recomendações para o Modelo de Referência

### Sincronismo de Mídias

#### Projeto MAESTRO: Autoria, Produção e Formatação de Documentos HiperMídia para TV Digital Interativa

<b>Código:</b>		<b>Versão:</b>	<i>01/01/00</i>
<b>Arquivo:</b>	<i>Recomendacao.pdf</i>	<b>Data de emissão:</b>	<i>07/12/05</i>
<b>Substitui o arquivo:</b>			
<b>Responsável:</b>	<b>Nome:</b>	<i>Luiz Fernando Gomes Soares</i>	
	<b>Fone:</b>	<i>0 21 3114-1500</i>	<b>e-mail:</b> <i>lfes@inf.puc-rio.br</i>
	<b>Instituição:</b>	<i>Pontifícia Universidade Católica do Rio de Janeiro</i>	



# SISTEMA BRASILEIRO DE TELEVISÃO DIGITAL

## Recomendações para o Modelo de Referência

### Sincronismo de Mídias

#### **Projeto MAESTRO: Autoria, Produção e Formatação de Documentos HiperMídia para TV Digital Interativa**

© Laboratório TeleMídia da PUC-Rio – Todos os direitos reservados

Impresso no Brasil

As informações contidas neste documento são de propriedade do Laboratório TeleMídia (PUC-Rio), sendo proibida a sua divulgação, reprodução ou armazenamento em base de dados ou sistema de recuperação sem permissão prévia e por escrito do Laboratório TeleMídia (PUC-Rio). As informações estão sujeitas a alterações sem notificação prévia.

Os nomes de produtos, serviços ou tecnologias eventualmente mencionadas neste documento são marcas registradas dos respectivos detentores.

Figuras apresentadas, quando obtidas de outros documentos, são sempre referenciadas e são de propriedade dos respectivos autores ou editoras referenciados.

Fazer cópias de qualquer parte deste documento para qualquer finalidade, além do uso pessoal, constitui violação das leis internacionais de direitos autorais.

Este documento foi publicado como Relatório Técnico de Pesquisa na série de Monografias do Departamento de Informática da PUC-Rio em dezembro de 2005, sob o número 41/05 e ISSN: 0103-9741.

**Laboratório TeleMídia**  
**Departamento de Informática**  
**Pontifícia Universidade Católica do Rio de Janeiro**  
Rua Marquês de São Vicente, 225, Prédio ITS - Gávea  
22453-900 – Rio de Janeiro – RJ – Brasil  
<http://www.telemidia.puc-rio.br>



## **RESPONSÁVEL PELO DOCUMENTO**

**Luiz Fernando Gomes Soares**

lfgs@inf.puc-rio.br

Lab. TeleMídia – Dep. de Informática – PUC-Rio  
R. Marquês de São Vicente, 225, ITS, Gávea  
22453-900, Rio de Janeiro, RJ, Brasil

## **Participantes da Elaboração da Recomendação**

Álvaro Veiga  
Antônio Tadeu Azevedo Gomes  
Bruno Feijó  
Bruno Maia da Cunha  
Bruno Santana da Silva  
Carla Faria Leitão  
Carlos de Salles Soares Neto  
Clarisse Sieckenius de Souza  
Fabio Wanderley Guerra  
Guilherme de Almeida Xavier  
Luiz Fernando Gomes Soares  
Marcelo Ferreira Moreno  
Márcio Ferreira Moreno  
Otávio A Martins Netto

Paula Salgado Lucena Rodrigues  
Paulo Badaró  
Rafael Ferreira Rodrigues  
Rafael Savignon Marinho  
Rodrigo Borges S. Santos  
Rodrigo Laiola Guimarães  
Romualdo Monteiro de Resende Costa  
Rogério Ferreira Rodrigues  
Rogério Miguel Coelho  
Simone Diniz Junqueira Barbosa  
Suzana Mesquita de Borba Maranhão  
Thaís Chiappetta Salgado  
Viviane Cristina Oliveira Aureliano

## Índice

1	Introdução .....	5
2	Vinte Questões sobre o Middleware declarativo .....	7
2.1	Por que middleware declarativo e middleware procedural? Um substitui o outro? .....	7
2.2	Qual o foco específico dos middlewares declarativos dos diversos sistemas de TV Digital existentes? .....	8
2.3	Sincronismo de mídias é um foco de grande importância? O que vem a ser? Qual a sua relação com a interatividade? .....	8
2.4	Por que o nome programa não-linear? .....	9
2.5	Qual a relação de programa não-linear com edição não-linear? .....	9
2.6	Quais as vantagens de se ter vários objetos de mídia compondo um programa não-linear (ou mesmo uma cena MPEG-4) e não termos o programa como um único objeto de vídeo?9	
2.7	Se o MPEG-4 já trabalha com vários objetos de mídia sincronizados, por que não se codifica tudo em MPEG-4? Qual a necessidade de uma outra linguagem no middleware declarativo? .....	10
2.8	Como os aplicativos para TV Digital se relacionam com o tipo de middleware? .....	10
2.9	Um programa de TV é considerado uma aplicação? .....	11
2.10	Por que não foi adotada uma linguagem declarativa com ênfase no sincronismo nos três sistemas de TV mais conhecidos? .....	11
2.11	Por que NCL para o middleware declarativo e não outras linguagens declarativas com foco no sincronismo? .....	12
2.12	O que será necessário no terminal de acesso para o suporte à NCL? .....	13
2.13	Se o Sistema Brasileiro adotar a NCL, ainda será possível a interoperabilidade com os outros sistemas que usam linguagens baseadas no HTML? .....	13
2.14	NCL dá suporte a objetos procedurais, como o HTML faz com scripts? .....	14
2.15	Se o Sistema Brasileiro adotar a NCL ainda será possível a interoperabilidade com os outros sistemas que poderão vir a adotar SMIL ou mesmo XMT-O? .....	14
2.16	É fácil desenvolver programas não-lineares usando NCL, comparativamente a outros sistemas? .....	14
2.17	Quais as ferramentas disponíveis para a edição de programas não-lineares usando NCL? 14	
2.18	Que diversas possibilidades de middleware ter-se-ia então para o SBTVD? .....	15
2.19	Programas não-lineares desenvolvidos em NCL poderiam ser exibidos em outros sistemas? .....	17
2.20	Resumidamente, por que NCL como linguagem e por que MAESTRO como Middleware declarativo do SBTVD? .....	17
3	Por que NCL como linguagem e Por que MAESTRO como Middleware declarativo do SBTVD? .....	18
	Referências .....	19



# 1 INTRODUÇÃO

Este documento traz as recomendações do consórcio formado para o cumprimento da RFP17 – Sincronismo de Mídias. Mais especificamente, ele traz as recomendações para a linguagem declarativa e o middleware declarativo a serem adotados no Sistema Brasileiro de TV Digital (SBTVD).

A proposição tem por base a linguagem declarativa NCL (Nested Context Language) [3], desenvolvida nos últimos 15 anos no Laboratório TeleMídia da PUC-Rio.

O middleware declarativo, denominado MAESTRO, tem como núcleo o formatador NCL [5]: módulo responsável pela orquestração da exibição sincronizada dos vários objetos de mídia que compõem uma aplicação declarativa, em particular, os programas não-lineares.

Como teste de conceito, validando esta proposição, foram implementadas e documentadas três versões do middleware declarativo: Maestro C/C++, Maestro C/C++ simplificada e Maestro Java.

A versão Maestro C/C++ foi implementada como o middleware *único* de um terminal de acesso de baixo custo, acoplando toda a parte do tratamento do fluxo de transporte desenvolvida em conjunto com o consórcio FlexTV (RFP 04).

Uma outra versão Maestro C/C++ simplificada foi desenvolvida em conjunto com o consórcio FlexTV (RFP 04) e integrada, como middleware declarativo, ao middleware procedural daquele consórcio.

A versão Maestro Java foi implementada como o middleware *declarativo* de um terminal de acesso mais sofisticado e também integrada ao middleware procedural do consórcio FlexTV (RFP 04).

Para validar a construção de programas não-lineares usando NCL, foi concebido e implementado um ambiente de autoria, também em Java [4], visando sua utilização pelos produtores de conteúdo em estações de trabalho. No editor, um conjunto de visões funciona de maneira integrada, permitindo que o autor tenha acesso a diferentes perspectivas do documento. Três visões gráficas, estrutural, temporal e leiaute, e uma visão textual compõem o editor.

Usando as ferramentas de edição, diversos programas não-lineares e aplicações declarativas foram desenvolvidas pelos parceiros do consórcio [6], validando a NCL, as ferramentas de edição e o formatador NCL.

Durante todo esse processo, fomos assistido por um grupo parceiro, responsável por avaliar a usabilidade de nossas ferramentas e programas [7], e propor melhorias.

Todos os resultados obtidos encontram-se documentados nos vários relatórios de produtos entregues pelo consórcio.

Tendo como base a vasta experiência do grupo e os trabalhos desenvolvidos neste projeto, podemos hoje propor, com absoluta certeza, o que pensamos ser melhor para o país, no que concerne à linguagem e ao middleware declarativo a serem adotados pelo SBTVD.

Escolhemos, para a apresentação de nossa recomendação, a forma didática de respostas a perguntas, através das quais encaminhamos o leitor para a conclusão final, justificando cada ponto da decisão.

Achamos melhor assim fazê-lo não só para sermos bem compreendidos, como também



para gerar um documento que pudesse ser de fácil entendimento, mesmo para os não especialistas.

Esperamos que este documento possa também ser útil na argumentação e convencimento dos diversos setores e atores participantes do processo decisório sobre o SBTVD.

Assim posto, o documento consiste de vinte perguntas, dezenove das quais respondidas na Seção 2.

A última questão é o resumo de tudo e aborda resumidamente o porquê de nossa recomendação. Por sua importância, a essa questão dedicamos a Seção 3.

## 2 VINTE QUESTÕES SOBRE O MIDDLEWARE DECLARATIVO

A principal pergunta para qual se busca resposta pode ser assim formulada: Por que NCL (Nested Context Language) como linguagem e por que MAESTRO como Middleware declarativo do SBTVD (Sistema Brasileiro de TV Digital)?

Para responder essa questão, temos de contextualizá-la, definindo alguns conceitos. Nesta seção, tentamos não nos aprofundar nos detalhes técnicos, que são discutidos em profundidade nas referências citadas (relatórios do projeto MAESTRO). Para a introdução dos conceitos, novas questões são formuladas e objetivamente respondidas.

### 2.1 Por que middleware declarativo e middleware procedural? Um substitui o outro?

*Não, um complementa o outro.*

Um middleware declarativo dá suporte a aplicações desenvolvidas em linguagens declarativas, ao passo que o procedural dá suporte a aplicações desenvolvidas em linguagens não declarativas.

Linguagens não declarativas podem seguir diferentes paradigmas. Tem-se assim as linguagens procedurais baseadas em módulos, orientadas a objetos etc. A literatura sobre TV digital, no entanto, cunhou o termo procedural para representar todas as linguagens que não são declarativas, ao contrário do usual na área de linguagens de programação. Linguagens procedurais comuns incluem Basic, C, Java, C++, Lua, ECMAScript etc.

Numa programação procedural devemos informar ao computador cada passo a ser executado. Pode-se afirmar que, em linguagens procedurais, o programador possui um maior poder sobre o código, sendo capaz de estabelecer todo o fluxo de controle e execução de seu programa. Entretanto, para isso, ele deve ser bem qualificado e conhecer bem os recursos de implementação.

Linguagens declarativas são linguagens de mais alto nível de abstração, usualmente ligadas a um domínio ou objetivo específico. Linguagens declarativas comuns incluem NCL, SMIL, HTML etc.

Nas linguagens declarativas, o programador fornece apenas o conjunto das tarefas a serem realizadas, não estando preocupado com os detalhes de como o executor da linguagem (interpretador, compilador ou a própria máquina real ou virtual de execução) realmente implementará essas tarefas. Linguagens declarativas resultam em uma declaração do resultado desejado, e, portanto, normalmente não necessitam de tantas linhas de código para definir uma certa tarefa.

*Em resumo, middlewares declarativos são mais adequados para aplicações cujo foco casa com o objetivo específico para o qual a linguagem foi desenvolvida, em caso contrário, o uso de middlewares procedurais é mais apropriado.*

Como na TV Digital os dois tipos de aplicações irão coexistir, é conveniente que o terminal de acesso integre os dois tipos de middleware.

Um aprofundamento da discussão sobre linguagens declarativas e procedurais pode ser obtido em [1].

## 2.2 Qual o foco específico dos middlewares declarativos dos diversos sistemas de TV Digital existentes?

O foco dos middlewares hoje existentes é a interatividade, isto é, eles têm como objetivo facilitar o desenvolvimento das aplicações com interação do usuário telespectador. Por isso, eles adotaram, como linguagem, derivações da linguagem HTML.

No entanto, esse foco é bastante restritivo. Para contornar suas restrições, as derivações de HTML, nos sistemas de TV usuais, permitem a inclusão de objetos especificados na linguagem procedural ECMAScript.

*Qualquer outro tipo de sincronismo de mídias, que não seja a interatividade, terá de ser descrito de forma procedural, ou seja, de forma mais complexa e mais longe dos objetivos do projetista da aplicação.*

## 2.3 Sincronismo de mídias é um foco de grande importância? O que vem a ser? Qual a sua relação com a interatividade?

A necessidade de sincronização em um Sistema de TV Digital está presente mesmo em sua aplicação mais primária: a exibição temporalmente sincronizada do fluxo de vídeo e áudio principal de um programa de TV.

Aplicações para TV digital muitas vezes devem lidar com a sincronização, espacial e temporal, de objetos de diferentes tipos de mídia, além dos objetos de vídeo e áudio que compõem o fluxo principal. Os vários objetos sincronizados (que são recebidos no chamado fluxo de dados) irão compor um documento multimídia. No caso mais geral, tal documento pode conter o fluxo de vídeo e áudio principal apenas como caso particular de seus objetos, sobre os quais, relacionamentos de sincronização podem ser estabelecidos. Tome como exemplo uma aplicação que, no momento preciso no qual o ator de um filme em exibição tira seu óculos, um vídeo-propaganda de uma ótica é exibido. Note que, nesse caso, o sincronismo foi baseado em um evento previsível do tempo (por exemplo, a retirada do óculos ocorre depois de transcorridos 10:30 minutos do início do filme).

Se o padrão MPEG-4 já tivesse uma maior penetração, certamente hoje sincronismo de mídias seria a palavra-chave. No MPEG-4, não existe a idéia de fluxo de vídeo e áudio principal, mas o conceito de cena, formada por vários objetos de mídia sincronizados no tempo e no espaço. A cena é de fato um documento multimídia.

Mas onde entra a interatividade? O sincronismo presente em documentos multimídia pode não ser apenas baseado em ocorrências (eventos) previsíveis, ou seja, em eventos onde é possível se prever o tempo e o local de suas ocorrências relativos a uma outra ocorrência (outro evento) qualquer. A interatividade do usuário é um exemplo de evento imprevisível.

*Note, assim, que a interatividade é apenas um caso particular, embora importante, de sincronismo de mídias.*

Documentos cujo fluxo normal de exibição pode ser alterado pela sincronização com outros eventos (previsíveis ou não previsíveis) são ditos não-lineares. *Programa não-linear é um caso particular de documentos não-lineares onde temos sincronizados vários objetos de mídia, incluindo entre eles o vídeo e o áudio principal (ou as cenas, e seus objetos, no caso do MPEG-4).*

*Provavelmente, programas não-lineares se constituirão na grande maioria das aplicações da TV Digital. Portanto, o sincronismo de mídias e não a interatividade deve ser o*

*foco das linguagens declarativas.*

## **2.4 Por que o nome programa não-linear?**

O termo vem em contraposição à forma seqüencial – linear – que caracteriza os programas para a TV analógica. Nesses últimos, existe um e apenas um caminho, seqüencial, de exibição. Ao contrário, os programas não-lineares são compostos de múltiplas cadeias de exibição, algumas exibidas em paralelo e outras como alternativas (ou seja, ou uma cadeia ou a outra) que dependem da escolha do usuário, do terminal onde o programa será exibido etc.

O exemplo mais simples de um programa não-linear é aquele onde, em um dado instante de exibição, o usuário telespectador pode escolher entre formas alternativas de continuação.

*Note assim que o programa deixa de poder ser representado por uma linha de tempo e passa a ter um fluxo de exibição que pode ser representado por um grafo.*

## **2.5 Qual a relação de programa não-linear com edição não-linear?**

*Nenhuma.*

A edição não-linear pode ser feita em um programa linear, como os da atual TV analógica. Ela consiste em permitir que certos trechos de um programa possam ser cortados ou colados sem ter que se esperar seqüencialmente a chegada de tal trecho.

Na edição não-linear, o vídeo do programa linear é, em geral, representado por uma linha de tempo (timeline), onde cortes de trechos ou colagens de trechos podem ser executados. Em programas não-lineares o paradigma de eixo de tempo (timeline) tem de ser abandonado, dada a imprevisibilidade das ações do usuário telespectador ou da escolha de alternativas de exibição.

## **2.6 Quais as vantagens de se ter vários objetos de mídia compondo um programa não-linear (ou mesmo uma cena MPEG-4) e não termos o programa como um único objeto de vídeo?**

Existem várias vantagens. Algumas são imperceptíveis para o usuário telespectador, mas são fundamentais para uma difusora de conteúdo, em termos de economia de banda passante e espaço de armazenamento. Entre essas vantagens podemos citar:

- A possibilidade de reuso do material audiovisual (sem necessidade de geração de cópia);
- A codificação de objetos podendo usar resoluções espaciais e temporais diferentes, que lhes forem mais apropriadas;
- A aplicação de técnicas de compactação e compressão mais apropriadas a cada objeto;
- A possibilidade de se ter objetos sintéticos participando da apresentação, junto com outros objetos de mídia natural (aqueles capturados por dispositivos como câmera, microfone etc.).

Outras vantagens são perceptíveis para o usuário telespectador e caracterizam a TV Digital. Entre elas podemos citar:

- A possibilidade de especificação de conteúdo alternativo para apresentações adaptativas (ao usuário, ao terminal de exibição etc.);

- A possibilidade de ações de interação do usuário, influenciando no fluxo de exibição do programa.

## **2.7 Se o MPEG-4 já trabalha com vários objetos de mídia sincronizados, por que não se codifica tudo em MPEG-4? Qual a necessidade de uma outra linguagem no middleware declarativo?**

É bem verdade que o MPEG-4 já possui uma linguagem para sincronização de todos os objetos de mídia que compõem uma cena de um programa. Mas só essa linguagem não basta, por várias razões.

Primeiro, porque muitas vezes se quer a sincronização com outros objetos de mídia que serão incorporados após a geração da codificação MPEG-4.

Segundo, porque a forma de sincronização MPEG-4 possui algumas restrições. Embora não exista restrições no relacionamento de sincronismo entre objetos da mesma cena, relacionamentos entre objetos de cenas diferentes são limitados. Por exemplo, objetos de cenas diferentes não podem ser exibidos simultaneamente. Isso vem da restrição que apenas uma cena pode ser exibida por vez.

Terceiro, a especificação de sincronismo através da linguagem BIFS do MPEG-4 não é trivial. Para contornar o problema, o padrão propõe o uso da linguagem XMT-O. XMT-O é derivada da linguagem SMIL e usa composições para definição de relações de sincronismo. O uso de composições é muito restritivo quando se quer definir relações de sincronismo mais complexas. Além disso, com composições é muito difícil estabelecer o sincronismo, entre objetos, em tempo de exibição (em exibições ao vivo). Por exemplo, se no momento de um gol quisermos disparar, sincronizadamente, uma imagem com as estatísticas do artilheiro, a solução em XMT-O não é trivial.

Dado o fato de que a geração de um programa não-linear deve ser simples a ponto de poder ser realizada por não especialistas em linguagens de programação para computadores, uma linguagem mais expressiva e amigável deve ser utilizada.

*Pelas razões mencionadas na pergunta 6, O MPEG-4 é um excelente padrão de codificação de informação audiovisual, mas precisa de uma linguagem de “cola” quando o objetos MPEG-4 são usados na composição de um programa não-linear. Essa linguagem deve ser suprida pelo middleware.*

## **2.8 Como os aplicativos para TV Digital se relacionam com o tipo de middleware?**

Nem todas as aplicações têm relação semântica com o conteúdo do áudio e vídeo principal, ou seja, com o significado da informação sendo exibida a partir do áudio e vídeo principal. A aplicação de correio eletrônico é um exemplo típico.

Outras aplicações têm relação semântica, mas não de sincronismo. Um exemplo é a aplicação que, em um programa sobre saúde, permite ao usuário telespectador realizar um teste de estresse a qualquer momento que desejar dentro do programa.

Os dois tipos de aplicações anteriormente mencionados não têm nenhuma relação de sincronismo com o áudio e vídeo principal, mas podem ser compostos por objetos de mídia cujas apresentações devam ser sincronizadas no tempo ou no espaço.

*Em geral, aplicações onde o sincronismo exerce papel preponderante são mais fáceis de serem especificadas usando uma linguagem declarativa. Porém, quando a exigência de sincronismo é apenas eventual, o melhor suporte é dado por uma linguagem procedural.*

*Um tipo de aplicação particular é aquela em que não só existe uma relação semântica entre seus objetos de mídia e o áudio e vídeo principal, mas também uma relação de sincronismo. Esse é exatamente o caso dos programas não-lineares. Na grande maioria desses casos, a linguagem declarativa tende a ser a preferencial.*

## **2.9 Um programa de TV é considerado uma aplicação?**

Exatamente. É a aplicação mais importante, mas é uma entre as várias aplicações que podem ser submetidas ao terminal de acesso.

## **2.10 Por que não foi adotada uma linguagem declarativa com ênfase no sincronismo nos três sistemas de TV mais conhecidos?**

Como já mencionamos, talvez a história fosse outra, caso o padrão de codificação de vídeo escolhido pelos sistemas, desde suas origens, fosse o MPEG-4.

No início da WWW, as redes ainda possuíam banda estreita e as informações transmitidas eram usualmente objetos de mídias discretas (texto e imagens). A interatividade do usuário imperava e, principalmente por sua simplicidade, a linguagem HTML se tornou um padrão de fato. Até os dias atuais HTML é uma das opções mais adequadas para aplicações puramente interativas. A necessidade de criação de páginas dinâmicas fez, contudo, com que a linguagem embutisse objetos procedurais em suas especificações: *scripts* e *applets*.

Com o aumento da capacidade de banda das redes e a melhora nas técnicas de codificação, os objetos de mídia contínua começaram a se tornar comuns. A princípio, a sincronização exigida por esses objetos, por serem simples, eram supridas pelos objetos procedurais embutidos na linguagem HTML.

Dada sua natureza de propósito geral, *scripts* e *applets* normalmente não apresentam mecanismos diretos para a especificação de sincronismo, exigindo mecanismos de mais baixo nível para a sincronização. Isso torna a programação mais flexível, porém mais complexa e suscetível a falhas. O nível de abstração oferecido para os autores é mais baixo, obrigando que os criadores de conteúdo trabalhem, na realidade, como programadores de software. É também difícil identificar pelo código as várias estruturas de sincronismo utilizadas pelo autor. Tudo isso torna difícil modularizar e manter uma apresentação.

Linguagens declarativas com ênfase no sincronismo se tornaram então mais conhecidas e candidatas a padrão para aplicações típicas de sincronismo de mídias na Web, como a linguagem SMIL, padrão W3C (World Wide Web Consortium).

Talvez pela sua simplicidade na definição da interação do usuário, HTML foi escolhida como a base do middleware declarativo dos três principais sistemas de TV Digital. Nota-se, no entanto, os primeiros ensaios de futuras mudanças. SMIL já foi proposta como linguagem para o MHP e NCL também vem sendo cogitada como opção para aquele middleware.

## 2.11 Por que NCL para o middleware declarativo e não outras linguagens declarativas com foco no sincronismo?

Além de NCL, são duas as outras opções mais naturais para um middleware declarativo: a linguagem SMIL e XMT-O. Contudo, é bom lembrarmos, mais uma vez, que nenhuma delas foi *ainda* formalmente adotada por nenhum dos sistemas de TV digital.

*Várias são as características de NCL não encontradas nas outras linguagens, o que nos leva à sua proposição como a linguagem do middleware declarativo do SBTVD. Vamos nos ater, resumidamente, nesta resposta, aos relacionamentos de sincronização, que são detalhados junto a outras características de NCL nas referências [1], [2] e [3].*

A resposta a esta questão é mais técnica e de maior dificuldade de compreensão, exigindo do leitor um conhecimento de linguagens declarativas para sincronismo, mas não podemos deixar de realçá-la.

Em SMIL e XMT-O, a representação de relacionamentos através de composições com semântica temporal embutida facilita bastante o trabalho do autor. Essa é a grande vantagem do uso de composições em SMIL e XMT-O. Por outro lado, nesse tipo de abordagem, relacionamentos mais complexos entre um conjunto de componentes têm que ser descritos por uma hierarquia de composições usando os tipos básicos. Isso nem sempre é desejável, pois obriga que a estrutura lógica do documento seja igual à sua estrutura de apresentação temporal. Essa estruturação pode se tornar ainda mais difícil de ser concebida quando existe um objeto mestre (o fluxo de áudio e vídeo principal) no qual se baseiam a grande maioria dos relacionamentos de sincronização.

A especificação de relacionamentos de sincronização espaço-temporais através de elos, que é a abordagem adotada por NCL, permite que o uso de composições possa ser dedicado à criação de relações de estruturação entre componentes de um documento. Além disso, o uso de elos para definir o sincronismo espaço-temporal permite que um autor defina qualquer tipo de relação de sincronização entre um grupo de componentes. Essa abordagem dá mais flexibilidade, já que o autor não precisa ficar limitado a alguns tipos de composição pré-definidos pela linguagem e pode estruturar seu documento de forma independente de suas relações de sincronização. Entretanto, para relacionamentos simples, pode ser mais trabalhosa a especificação utilizando elos.

Dadas as vantagens e desvantagens do uso de composições e elos para representar relações, o ideal é oferecer ao autor todas as possibilidades: o uso de composições para estruturação lógica, o uso de elos para especificar sincronização e o uso de composições com semântica, por exemplo temporal, fazendo a estrutura lógica casar com a estrutura de apresentação, quando apropriado. Melhor ainda se a linguagem de autoria permitir a criação de tipos de composição com a semântica desejada pelo autor e que possam ser utilizados (e reutilizados) de acordo com sua necessidade. Assim, a estrutura de apresentação de um documento pode se adaptar à sua estrutura lógica, e não o contrário. A linguagem NCL é a única linguagem que oferece ao autor essas várias possibilidades, através do conceito de template de composição. Dessa forma, tipos pré-definidos de composições, como os oferecidos pela linguagem SMIL, podem ser vistos como templates. Através do módulo XTemplate da linguagem NCL, novos templates de composição podem ser criados, generalizando os tipos de composição que a linguagem de autoria oferece.

Elos (relacionamentos) NCL podem ser multiponto (múltiplos participantes da relação) e representar relações com semântica causal ou de restrição. Além disso, uma mesma relação pode relacionar eventos de diversos tipos (espaciais e temporais), além dos tradicionais eventos

temporais, normalmente contemplados pelas outras linguagens. SMIL e XMT-O só oferecem elos ponto-a-ponto, que podem ser disparados por eventos temporais (incluindo a interatividade). A possibilidade de usar eventos na especificação temporal de um documento SMIL deu bastante flexibilidade à linguagem, comparada com sua versão anterior. Entretanto, para especificar relacionamentos complexos, o autor deve mesclar o uso dos contêineres temporais (par, seq e excl) com o uso de eventos, e ainda com o uso de elos, se for o caso, para especificar relacionamentos que envolvem a ocorrência de vários tipos de evento. Além disso, a utilização de eventos, para definição de relacionamentos, pode inviabilizar a grande vantagem de SMIL de se ter composições com semânticas bem definidas: as composições seq e par podem ter suas semânticas alteradas pelo uso de eventos internos a elas. Em NCL, uma relação, por mais complexa que seja, pode ser abstraída por um único elemento (denominado conector) e utilizada da mesma forma que uma relação bem simples.

Merece também destaque a especificação flexível do comportamento temporal de objetos em NCL. Desde a versão 1.0 de NCL, as durações de objetos na especificação de um documento podem ser especificadas de forma flexível (dentro de um intervalo de tolerância). Essa facilidade só foi incluída na linguagem SMIL em sua segunda versão. Além disso, NCL prevê a especificação de funções de custo para guiar o formatador em possíveis ajustes temporais durante a apresentação do documento, visando minimizar a perda de sincronismo, facilidade que não é encontrada nas linguagens SMIL e XMT-O.

Finalmente, eventos de sincronismo gerados em tempo de exibição dos programas (autoria de programas não-lineares ao vivo) são muito mais naturais de serem representados por elos. O uso de composições, nesse caso, além de complicado, desestrutura totalmente o programa, caso se queira armazená-lo para uma posterior exibição.

## **2.12 O que será necessário no terminal de acesso para o suporte à NCL?**

*No terminal de acesso será necessário um orquestrador de apresentação, denominado formatador NCL, núcleo central do middleware declarativo MAESTRO.*

O formatador tem a função de receber a especificação NCL, juntamente com os vários objetos de mídia, e orquestrar a exibição dos objetos de acordo com a especificação de sincronização dada pela descrição em NCL.

No tempo preciso especificado, o formatador inicia o exibidor de cada objeto, passando-lhe os dados a exibir.

## **2.13 Se o Sistema Brasileiro adotar a NCL, ainda será possível a interoperabilidade com os outros sistemas que usam linguagens baseadas no HTML?**

*É possível sim.*

NCL permite vários tipos de objetos de mídia, além dos tradicionais vídeo, áudio, texto e imagem. Páginas HTML, com scripts embutidos ou não, são tratados como um objeto NCL. Ao ter de exibir um documento HTML, o formatador NCL instancia um navegador HTML passando-lhe os dados necessários.

*O navegador HTML é apenas um entre vários exibidores de objetos de mídia NCL.*

## 2.14 NCL dá suporte a objetos procedurais, como o HTML faz com scripts?

*A resposta é sim.*

ECMAScripts podem ser embutidos em páginas HTML e resolvidos pelo navegador HTML. ECMAScripts podem também ser instanciados diretamente pelo formatador NCL, interagindo, neste caso, com o formatador.

LuaScripts podem também ser instanciados diretamente pelo formatador, bem como aplicações Java (NCLets: Xlets como objetos NCL).

Tudo isso é possível desde que as respectivas máquinas de execução sejam implementadas no middleware declarativo juntamente com o formatador e os outros exibidores de mídia. Em outras palavras, cada máquina é encarada apenas como mais um exibidor do formatador e o programa procedural como apenas mais um tipo de objeto NCL.

## 2.15 Se o Sistema Brasileiro adotar a NCL ainda será possível a interoperabilidade com os outros sistemas que poderão vir a adotar SMIL ou mesmo XMT-O?

Mais uma vez, a resposta é sim.

*NCL já tem conversores desenvolvidos para a linguagem SMIL e XMT-O, e vice-versa.*

## 2.16 É fácil desenvolver programas não-lineares usando NCL, comparativamente a outros sistemas?

A curva de aprendizado de NCL, para explorá-la em toda sua potencialidade, é substancial.

É fácil e rápido desenvolver programas simples em NCL. O mesmo pode ser dito de programas complexos, mas depois que o usuário já tiver adquirido prática e disponha de exemplos e bibliotecas de onde possa fazer reuso de partes da especificação, o que é incentivado na linguagem.

Comparativamente, entretanto, é mais fácil produzir programas não-lineares em NCL do que através de ECMAScripts embutidos em especificações HTML. Para programas simples, o uso de SMIL e XMT-O pode ser mais fácil quando não se explora as potencialidades dos *templates* NCL. No entanto, para programas lineares de média a alta complexidade, para produções ao vivo e quando se dispõe de uma biblioteca para reuso, a produção em NCL tem se mostrado mais simples e mais adequada.

## 2.17 Quais as ferramentas disponíveis para a edição de programas não-lineares usando NCL?

*Para o ambiente de autoria na linguagem NCL foi desenvolvido o editor S2T2View.*

No editor, um conjunto de visões gráficas funciona de maneira integrada, para permitir diferentes perspectivas do documento, dependendo do foco de criação/edição. As ferramentas gráficas são baseadas em três visões: estrutural, temporal e leiaute, conforme detalhado em [4].

Além das visões gráficas, o ambiente de autoria integra uma visão textual dos

documentos. A visão textual também funciona integrada com as visões gráficas e possui facilidades como validação sintática dos documentos, destaque de palavras-chave da linguagem e filtros para melhor orientação do autor em documentos mais complexos.

## 2.18 Que diversas possibilidades de middleware ter-se-ia então para o SBTVD?

*Existem várias combinações possíveis para os diversos módulos que compõem o middleware (declarativo + procedural) do SBTVD.*

Entre as várias alternativas possíveis, três merecem destaque.

A primeira, apresentada na Figura 1, propõe um middleware apenas com sua parte declarativa, embora aplicações procedurais possam ser lançadas a partir de aplicações declarativas. No entanto, essas aplicações procedurais devem ser bastante simples.

Guias eletrônicos de programação, navegador Web, enfim, aplicações onde o sincronismo (incluindo a interatividade) é o foco são privilegiadas nessa alternativa. O destaque dessas aplicações vai para os programas não-lineares.

Essa alternativa 1 exige um poder computacional pequeno dos terminais de acesso. Incluindo todas as bibliotecas das máquinas virtuais e exibidores, ela ocupa um espaço de 1.3 Mbytes. O protótipo (teste de conceito) já em funcionamento (desenvolvido em C/C++) executa em um processador AMD i586 266 MHz, com 32M de RAM. É a alternativa proposta para os terminais de acesso de baixo custo.

*É importante frisarmos que essa alternativa não apenas exhibe aplicações NCL, mas qualquer conteúdo declarativo gerado nos sistemas americano, europeu e japonês.*

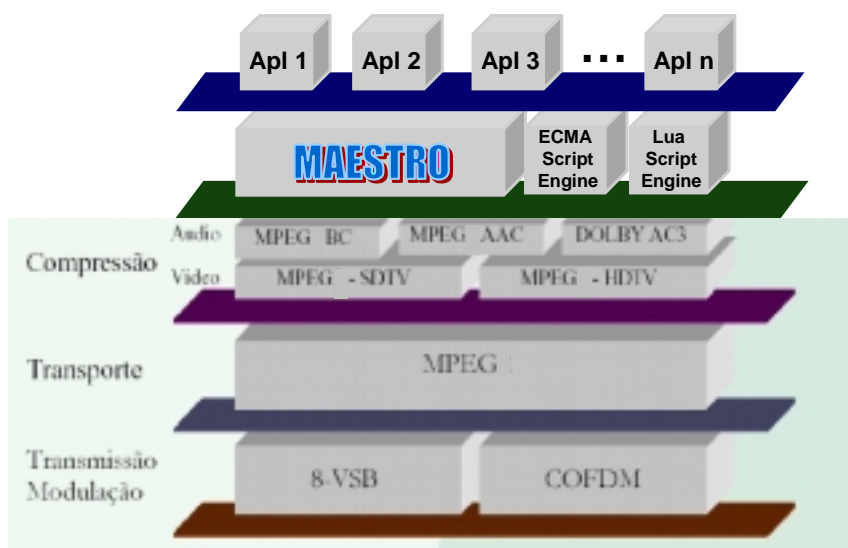


Figura 1: Middleware alternativo de baixo custo (alternativa 1)

A segunda alternativa prevê a integração do middleware declarativo MAESTRO com o middleware procedural. O protótipo (teste de conceito), já em funcionamento, integra duas versões do MAESTRO (em C++ e Java) ao middleware procedural FlexTV, conforme ilustra a Figura 2. A versão Java do formatador NCL ocupa um espaço de 370 Kbytes, que deve ser adicionado ao espaço exigido pelo middleware procedural FlexTV. A versão C++ ocupa um

espaço de 900 Kbytes.

Nesta segunda alternativa, aplicações procedurais podem ser lançadas a partir de aplicações declarativas e vice-versa. Aplicações procedurais e declarativas poderão coexistir, ampliando o leque de aplicações do Terminal de Acesso.

Obviamente, essa alternativa vai exigir um terminal de acesso de maior capacidade, mas será capaz de dar suporte a qualquer tipo de aplicação.

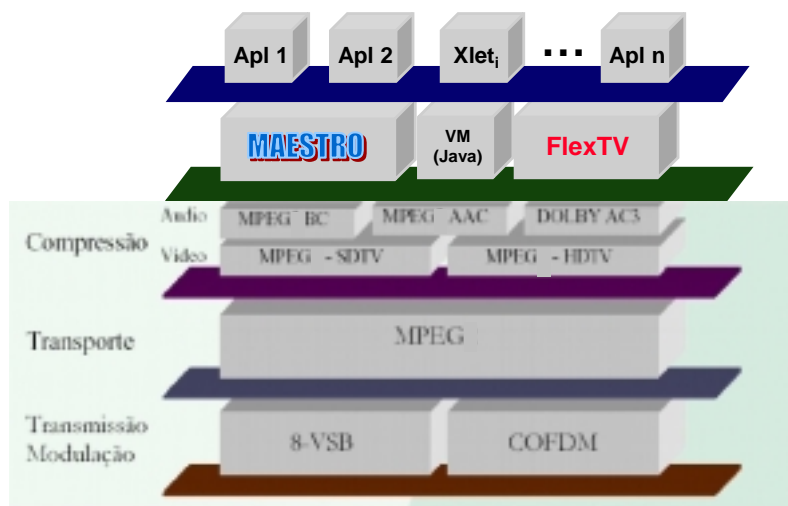


Figura 2: Middleware declarativo e procedural residentes (alternativa 2)

A terceira alternativa difere da segunda, por não ter o middleware declarativo residente.

Implementado como um XLet, o formatador NCL poderá ser carregado, via fluxo de dados, não apenas no middleware do SBTVD, conforme apresenta a Figura 3, mas em qualquer outro Sistema de TV Digital. A versão XLet do formatador NCL ocupa um espaço de 370 Kbytes.

O protótipo dessa alternativa, em concepção, prevê a implementação do formatador NCL baseado em componentes. Dessa forma, apenas os componentes necessários à exibição de um programa declarativo particular precisarão ser carregados, antes da execução do programa.

*É importante frisarmos que essa alternativa não apenas exhibe qualquer conteúdo declarativo gerado nos sistemas americano, europeu e japonês, como também torna possível a exibição de aplicações NCL em qualquer um desses sistemas.*

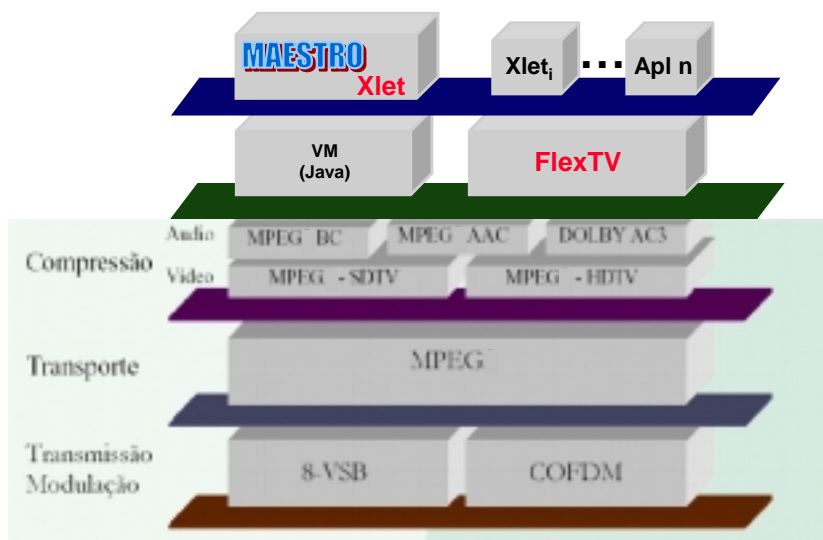


Figura 3: Middleware declarativo XLet e procedural residente (alternativa 3)

*Concluindo, é importante salientarmos que todas as 3 alternativas podem coexistir. Podemos ter terminais de acesso usando qualquer das alternativas, variando em custo e capacidade.*

## 2.19 Programas não-lineares desenvolvidos em NCL poderiam ser exibidos em outros sistemas?

As arquiteturas apresentadas na pergunta 2.18 já responderam a esta questão, mas é importante salientarmos mais uma vez que:

- *Qualquer conteúdo declarativo gerado nos sistemas americano, europeu e japonês terão suporte no middleware declarativo MAESTRO, bem como, obviamente, conteúdos desenvolvidos em NCL.*
- *A implementação do formatador NCL em XLet permite que aplicações NCL possam ser executadas em qualquer um dos sistemas: americano, europeu ou japonês.*

## 2.20 Resumidamente, por que NCL como linguagem e por que MAESTRO como Middleware declarativo do SBTVD?

Agora sim, podemos responder a essa questão, resumindo tudo o que foi discutido nas respostas às 19 questões anteriores. Pela importância da resposta, vamos reservar a ela a próxima seção.

### **3 POR QUE NCL COMO LINGUAGEM E POR QUE MAESTRO COMO MIDDLEWARE DECLARATIVO DO SBTVD?**

Resumida e objetivamente:

- Porque MAESTRO é compatível com middlewares declarativos de todos os outros sistemas. Todos os conteúdos declarativos produzidos naqueles sistemas podem ser exibidos no MAESTRO;
- Porque MAESTRO usa NCL, a linguagem declarativa de maior poder de expressão para definição da estruturação e sincronismo entre objetos de mídia;
- Porque MAESTRO é a evolução natural dos middlewares, usa uma linguagem (NCL) mais apropriada por ter como foco não apenas a interatividade, mas o sincronismo de mídias;
- Porque NCL é inovadora. É fruto de 15 anos de pesquisa cujos resultados são internacionalmente reconhecidos;
- Porque MAESTRO e NCL foram totalmente projetados por pesquisadores brasileiros e, por isso, o Brasil tem o domínio da tecnologia;
- Porque o país, tendo o domínio da tecnologia, pode alterá-la ou ajustá-la quando for da conveniência de sua indústria de conteúdo ou de equipamentos;
- Porque o país, tendo o domínio da tecnologia, pode usá-la para fomentar novos negócios, novas indústrias e a geração de empregos qualificados;
- Porque MAESTRO e NCL são produtos nacionais, cujas ferramentas (editor e formatador) estão disponíveis como software livre, e nenhuma taxa será paga a nenhum outro país, ou empresa de outro país, por suas utilizações;
- Porque MAESTRO é escalável e possibilita a construção, desde terminais de acesso de baixo custo, atendendo à população de baixa renda, a terminais de acesso complexos;
- Porque MAESTRO é produto de exportação; é um componente de software que pode ser integrado em quaisquer dos principais sistemas de TV digital existentes;
- Porque conteúdos em NCL podem ser exportados para qualquer país usuário dos três principais sistemas de TV digital;
- Porque ferramentas de edição NCL são softwares que poderão ser exportados para o desenvolvimento de conteúdo NCL, visando qualquer dos três principais sistemas existentes;
- Porque NCL é produto de exportação que pode introduzir um diferencial aos conteúdos produzidos no Brasil, em particular os programas não-lineares.

## REFERÊNCIAS

- [01] Soares L.F.G., Rodrigues R.F. “Levantamento do Estado da Arte em Sincronismo de Mídias”. Relatório Técnico de Pesquisa da série de Monografias do Departamento de Informática da PUC-Rio. No. 12. abril de 2005. ISSN: 0103-9741.
- [02] Soares L.F.G., Rodrigues R.F. “Nested Context Model 3.0: Part 1 – NCM Core”. Relatório Técnico de Pesquisa da série de Monografias do Departamento de Informática da PUC-Rio. No. 12. abril de 2005. ISSN: 0103-9741.
- [03] Soares L.F.G., Rodrigues R.F. “Nested Context Model 3.0. Part 5 – NCL (Nested Context Language)”. Relatório Técnico de Pesquisa da série de Monografias em Ciência da Computação do Departamento de Informática, PUC-Rio, No. 26/05. Rio de Janeiro. Junho de 2005. ISSN 0103-9741.
- [04] Produto 3: Autoria de Documentos Hipermídia para TV Digital Interativa. RFP-17 - Projeto MAESTRO: Autoria, Produção e Formatação de Documentos Hipermídia para TV Digital Interativa.
- [05] Produto 4: Mecanismo para Sincronismo de Mídias. RFP-17 - Projeto MAESTRO: Autoria, Produção e Formatação de Documentos Hipermídia para TV Digital Interativa.
- [06] Produto 5: Concepção Artística de Programas Audio-Visuais Interativos. RFP-17 - Projeto MAESTRO: Autoria, Produção e Formatação de Documentos Hipermídia para TV Digital Interativa.
- [07] Produto 6: *Metodologia de Avaliação do Mecanismo de Sincronismo de Mídias*. RFP-17 - Projeto MAESTRO: Autoria, Produção e Formatação de Documentos Hipermídia para TV Digital Interativa.